Load The Dataset (Week 2)

In [1]:
```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

#ingest data
df = pd.read_csv('https://raw.githubusercontent.com/Christine971224/Analytics-2023/mast
df.head()
```

Out[1]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival
0	Resort Hotel	0	342	2015	July	27	
1	Resort Hotel	0	737	2015	July	27	
2	Resort Hotel	0	7	2015	July	27	
3	Resort Hotel	0	13	2015	July	27	
4	Resort Hotel	0	14	2015	July	27	

5 rows × 36 columns

In [2]:
```
#basic information of dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 36 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   hotel                           119390 non-null  object
 1   is_canceled                     119390 non-null  int64
 2   lead_time                       119390 non-null  int64
 3   arrival_date_year               119390 non-null  int64
 4   arrival_date_month              119390 non-null  object
 5   arrival_date_week_number        119390 non-null  int64
 6   arrival_date_day_of_month       119390 non-null  int64
 7   stays_in_weekend_nights         119390 non-null  int64
 8   stays_in_week_nights            119390 non-null  int64
 9   adults                          119390 non-null  int64
 10  children                        119386 non-null  float64
```

```
 11  babies                          119390 non-null  int64
 12  meal                            119390 non-null  object
 13  country                         118902 non-null  object
 14  market_segment                  119390 non-null  object
 15  distribution_channel            119390 non-null  object
 16  is_repeated_guest               119390 non-null  int64
 17  previous_cancellations          119390 non-null  int64
 18  previous_bookings_not_canceled  119390 non-null  int64
 19  reserved_room_type              119390 non-null  object
 20  assigned_room_type              119390 non-null  object
 21  booking_changes                 119390 non-null  int64
 22  deposit_type                    119390 non-null  object
 23  agent                           103050 non-null  float64
 24  company                         6797 non-null    float64
 25  days_in_waiting_list            119390 non-null  int64
 26  customer_type                   119390 non-null  object
 27  adr                             119390 non-null  float64
 28  required_car_parking_spaces     119390 non-null  int64
 29  total_of_special_requests       119390 non-null  int64
 30  reservation_status              119390 non-null  object
 31  reservation_status_date         119390 non-null  object
 32  name                            119390 non-null  object
 33  email                           119390 non-null  object
 34  phone-number                    119390 non-null  object
 35  credit_card                     119390 non-null  object
dtypes: float64(4), int64(16), object(16)
memory usage: 32.8+ MB
```

In [3]:

```
df.isnull().mean()
```

Out[3]:

```
hotel                             0.000000
is_canceled                       0.000000
lead_time                         0.000000
arrival_date_year                 0.000000
arrival_date_month                0.000000
arrival_date_week_number          0.000000
arrival_date_day_of_month         0.000000
stays_in_weekend_nights           0.000000
stays_in_week_nights              0.000000
adults                            0.000000
children                          0.000034
babies                            0.000000
meal                              0.000000
country                           0.004087
market_segment                    0.000000
distribution_channel              0.000000
is_repeated_guest                 0.000000
previous_cancellations            0.000000
previous_bookings_not_canceled    0.000000
reserved_room_type                0.000000
assigned_room_type                0.000000
booking_changes                   0.000000
deposit_type                      0.000000
agent                             0.136862
company                           0.943069
days_in_waiting_list              0.000000
customer_type                     0.000000
adr                               0.000000
required_car_parking_spaces       0.000000
```

```
total_of_special_requests       0.000000
reservation_status              0.000000
reservation_status_date         0.000000
name                            0.000000
email                           0.000000
phone-number                    0.000000
credit_card                     0.000000
dtype: float64
```

In [4]:

```
# adults, babies and children can't be zero at same time, so dropping the rows having a
filter = (df.children == 0) & (df.adults == 0) & (df.babies == 0)
df[filter]
```

Out[4]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number
2224	Resort Hotel	0	1	2015	October	41
2409	Resort Hotel	0	0	2015	October	42
3181	Resort Hotel	0	36	2015	November	47
3684	Resort Hotel	0	165	2015	December	53
3708	Resort Hotel	0	165	2015	December	53
...	...	...	...	...	...	...
115029	City Hotel	0	107	2017	June	26
115091	City Hotel	0	1	2017	June	26
116251	City Hotel	0	44	2017	July	28
116534	City Hotel	0	2	2017	July	28
117087	City Hotel	0	170	2017	July	30

180 rows × 36 columns

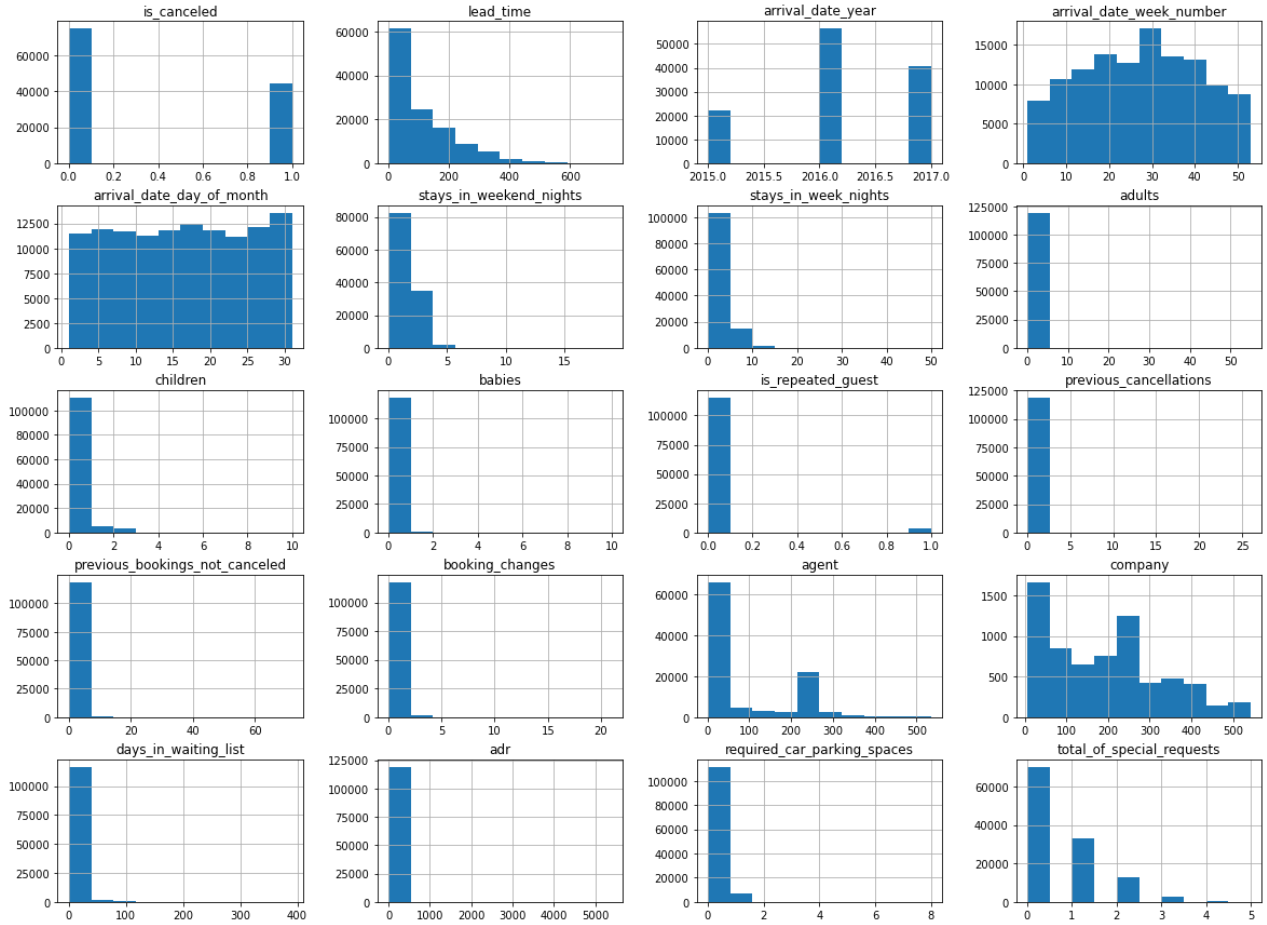
In [5]:
```
# transpose the resulting DataFrame
df.describe([0.01,0.05,0.1,0.25,0.5,0.75,0.99]).T
```

Out[5]:

	count	mean	std	min	1%	5%	10%	2
is_canceled	119390.0	0.370416	0.482918	0.00	0.0	0.0	0.0	0
lead_time	119390.0	104.011416	106.863097	0.00	0.0	0.0	3.0	18
arrival_date_year	119390.0	2016.156554	0.707476	2015.00	2015.0	2015.0	2015.0	2016
arrival_date_week_number	119390.0	27.165173	13.605138	1.00	2.0	5.0	8.0	16
arrival_date_day_of_month	119390.0	15.798241	8.780829	1.00	1.0	2.0	4.0	8
stays_in_weekend_nights	119390.0	0.927599	0.998613	0.00	0.0	0.0	0.0	0
stays_in_week_nights	119390.0	2.500302	1.908286	0.00	0.0	0.0	1.0	1
adults	119390.0	1.856403	0.579261	0.00	1.0	1.0	1.0	2
children	119386.0	0.103890	0.398561	0.00	0.0	0.0	0.0	0
babies	119390.0	0.007949	0.097436	0.00	0.0	0.0	0.0	0
is_repeated_guest	119390.0	0.031912	0.175767	0.00	0.0	0.0	0.0	0
previous_cancellations	119390.0	0.087118	0.844336	0.00	0.0	0.0	0.0	0
previous_bookings_not_canceled	119390.0	0.137097	1.497437	0.00	0.0	0.0	0.0	0
booking_changes	119390.0	0.221124	0.652306	0.00	0.0	0.0	0.0	0
agent	103050.0	86.693382	110.774548	1.00	1.0	1.0	6.0	9
company	6797.0	189.266735	131.655015	6.00	16.0	40.0	40.0	62
days_in_waiting_list	119390.0	2.321149	17.594721	0.00	0.0	0.0	0.0	0
adr	119390.0	101.831122	50.535790	-6.38	0.0	38.4	50.0	69
required_car_parking_spaces	119390.0	0.062518	0.245291	0.00	0.0	0.0	0.0	0
total_of_special_requests	119390.0	0.571363	0.792798	0.00	0.0	0.0	0.0	0

In [6]:
```
import matplotlib.pyplot as plt

# generate histograms for all the columns
df.hist(figsize=(20,15))
plt.show()
```

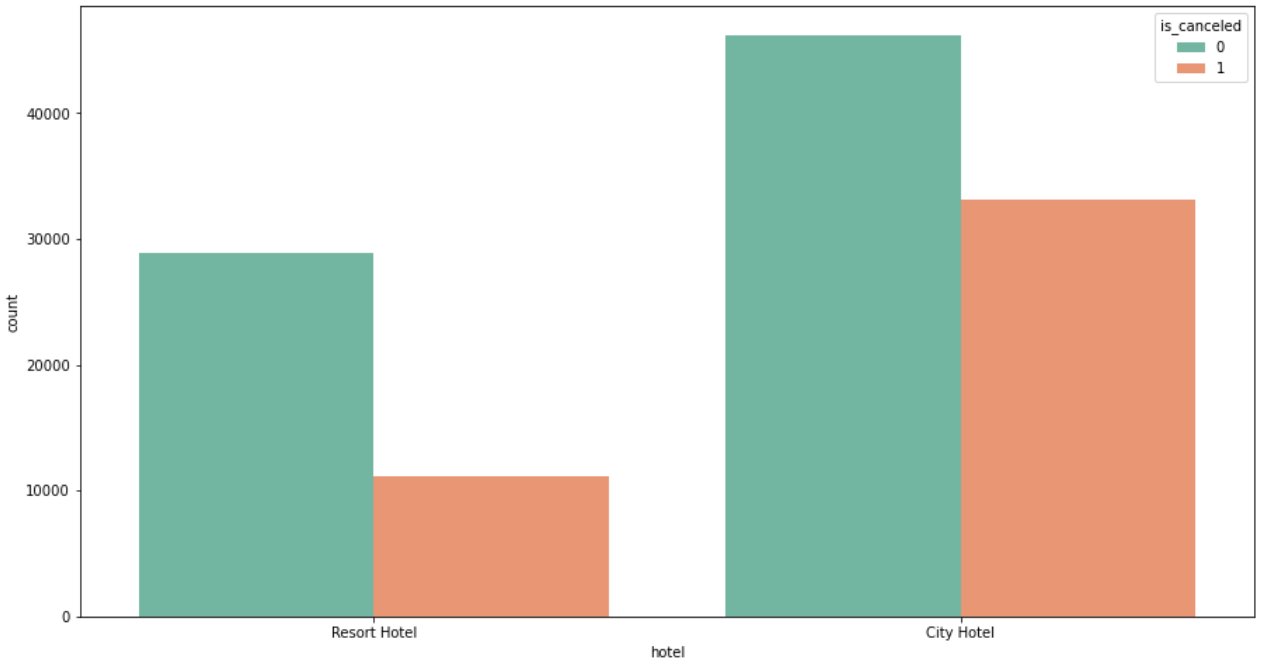


EDA（Week 3）

1. Hotel bookings and cancellations

In [7]:
```
# The number of hotel reservations and cancellations can directly show the actual numbe
import seaborn as sns
plt.figure(figsize=(15,8))
sns.countplot(x='hotel'
              ,data=df
              ,hue='is_canceled'
              ,palette=sns.color_palette('Set2',2)
             )
```

Out[7]:
```
<AxesSubplot:xlabel='hotel', ylabel='count'>
```



In [8]:
```
#calculate the proportion of cancellations for each unique value in the 'hotel' column
hotel_cancel=(df.loc[df['is_canceled']==1]['hotel'].value_counts()/df['hotel'].value_co
print('Hotel cancellations'.center(20),hotel_cancel,sep='\n')
```

```
Hotel cancellations
City Hotel      0.417270
Resort Hotel    0.277634
Name: hotel, dtype: float64
```

Comment: City Hotel's booking volume and cancellation volume are both higher than Resort Hotel's, but Resort Hotel's cancellation rate is 27.8%, while City Hotel's cancellation rate reaches 41.7%.

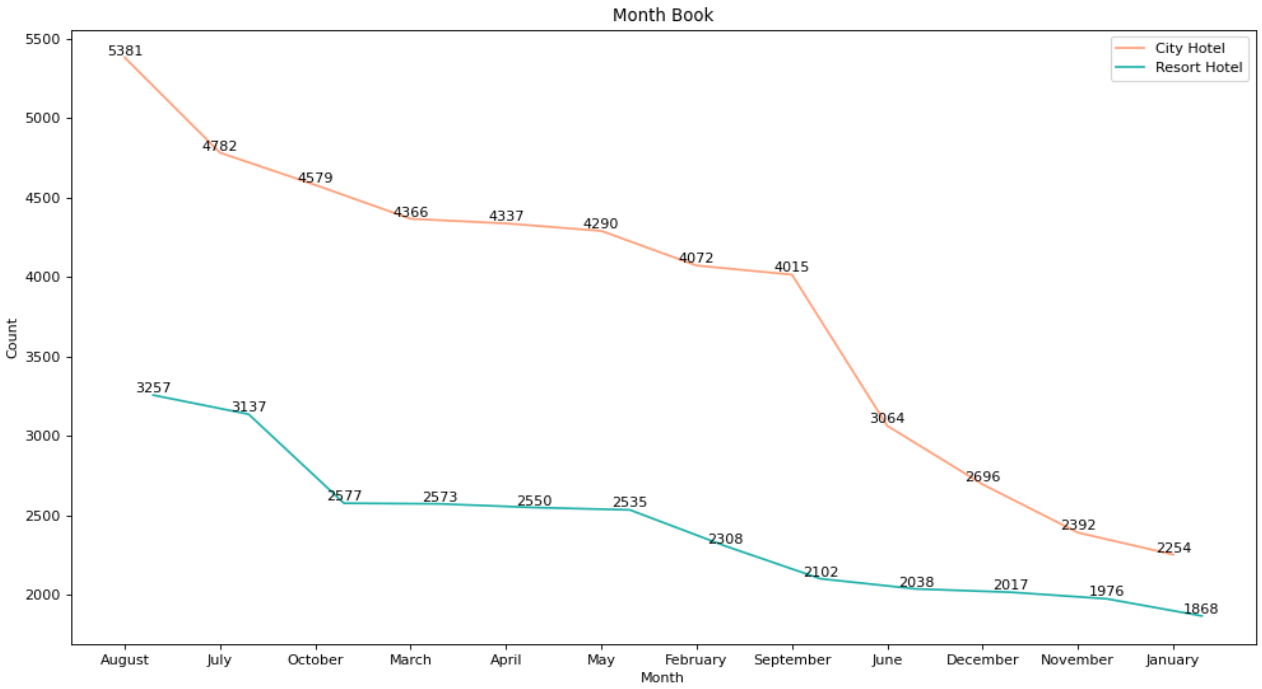
1. Hotel bookings by month

In [9]:
```
#create a plot to visualize the number of bookings for "City Hotel" and "Resort Hotel"
city_hotel=df[(df['hotel']=='City Hotel') & (df['is_canceled']==0)]
resort_hotel=df[(df['hotel']=='Resort Hotel') & (df['is_canceled']==0)]
for i in [city_hotel,resort_hotel]:
    i.index=range(i.shape[0])

city_month=city_hotel['arrival_date_month'].value_counts()
resort_month=resort_hotel['arrival_date_month'].value_counts()
name=resort_month.index
x=list(range(len(city_month.index)))
y=city_month.values
x1=[i+0.3 for i in x]
y1=resort_month.values
width=0.3
plt.figure(figsize=(15,8),dpi=80)
plt.plot(x,y,label='City Hotel',color='lightsalmon')
plt.plot(x1,y1,label='Resort Hotel',color='lightseagreen')
plt.xticks(x,name)
plt.legend()
plt.xlabel('Month')
```

```
plt.ylabel('Count')
plt.title('Month Book')
for x,y in zip(x,y):
    plt.text(x,y+0.1,'%d' % y,ha = 'center',va = 'bottom')

for x,y in zip(x1,y1):
    plt.text(x,y+0.1,'%d' % y,ha = 'center',va = 'bottom')
```



Comment: Peak booking months are August and July. Preliminary judgment is that the long holiday caused the peak period.

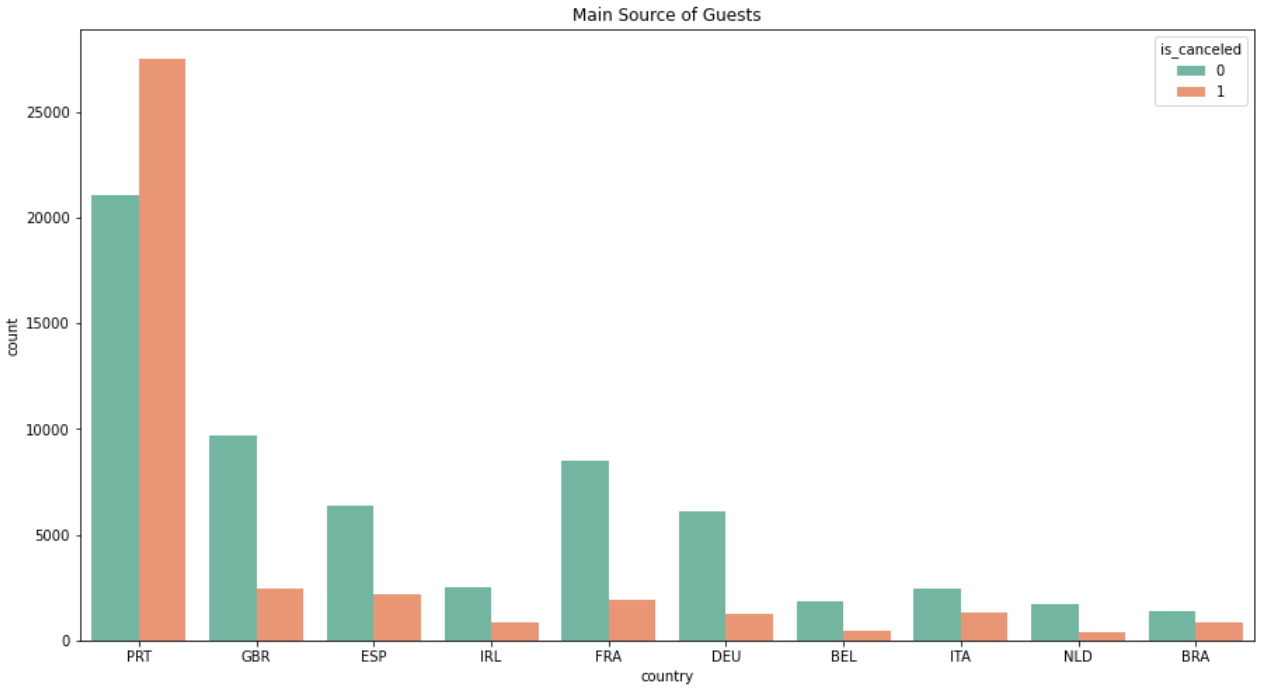
1. Customer origin and booking cancellation rate

In [10]:

```
#create a plot using Seaborn's countplot to visualize the top 10 countries from which b
country_book=df['country'].value_counts()[:10]
country_cancel=df[(df.country.isin (country_book.index)) & (df.is_canceled==1)]['countr
plt.figure(figsize=(15,8))
sns.countplot(x='country'
              ,data=df[df.country.isin (country_book.index)]
              ,hue='is_canceled'
              ,palette=sns.color_palette('Set2',2)
             )
plt.title('Main Source of Guests')
```

Out[10]:

```
Text(0.5, 1.0, 'Main Source of Guests')
```



In [11]:
```
#calculate the cancellation rate for each of the top 10 countries (those with the highe
country_cancel_rate=(country_cancel/country_book).sort_values(ascending=False)
print('Customer cancellation rates by country'.center(10),country_cancel_rate,sep='\n')
```

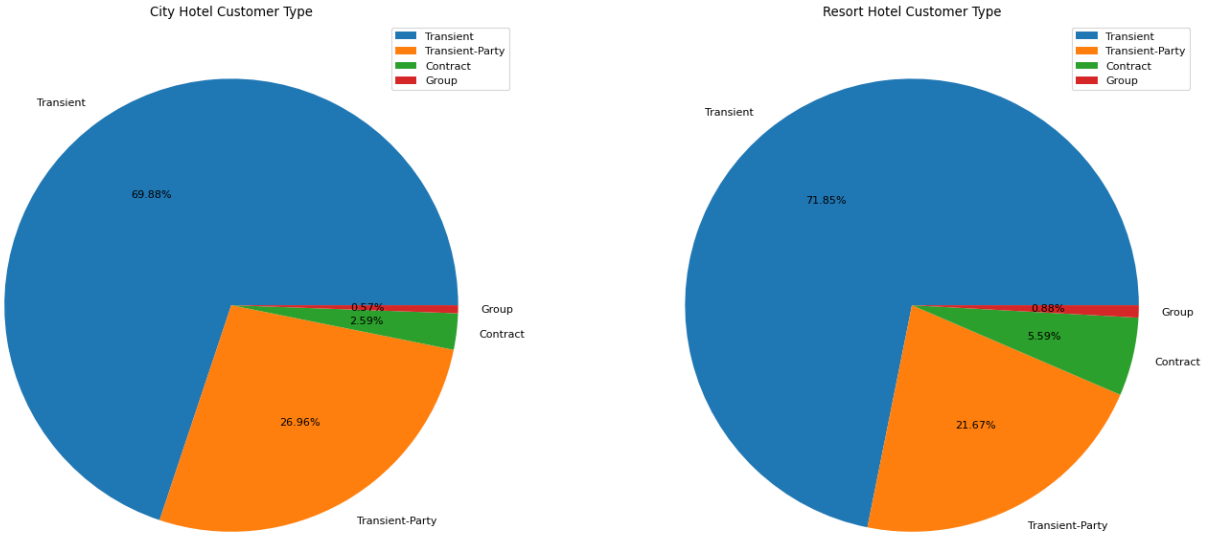
```
Customer cancellation rates by country
PRT    0.566351
BRA    0.373201
ITA    0.353956
ESP    0.254085
IRL    0.246519
BEL    0.202391
GBR    0.202243
FRA    0.185694
NLD    0.183935
DEU    0.167147
Name: country, dtype: float64
```

The peak season for both Resort hotel and City hotel is July and August in summer, and the main sources of tourists are European countries. This is in line with the characteristics of European tourists who prefer summer travel. It is necessary to focus on countries with high cancellation rates such as Portugal (PRT) and the United Kingdom (BRT). Main source of customers.

1. Customer type

In [12]:
```
#visualize the distribution of customer types for two types of hotels: City Hotel and R
city_customer=city_hotel.customer_type.value_counts()
resort_customer=resort_hotel.customer_type.value_counts()
plt.figure(figsize=(21,12),dpi=80)
plt.subplot(1,2,1)
plt.pie(city_customer,labels=city_customer.index,autopct='%.2f%%')
plt.legend(loc=1)
plt.title('City Hotel Customer Type')
plt.subplot(1,2,2)
plt.pie(resort_customer,labels=resort_customer.index,autopct='%.2f%%')
```

```
plt.title('Resort Hotel Customer Type')
plt.legend()
plt.show()
```

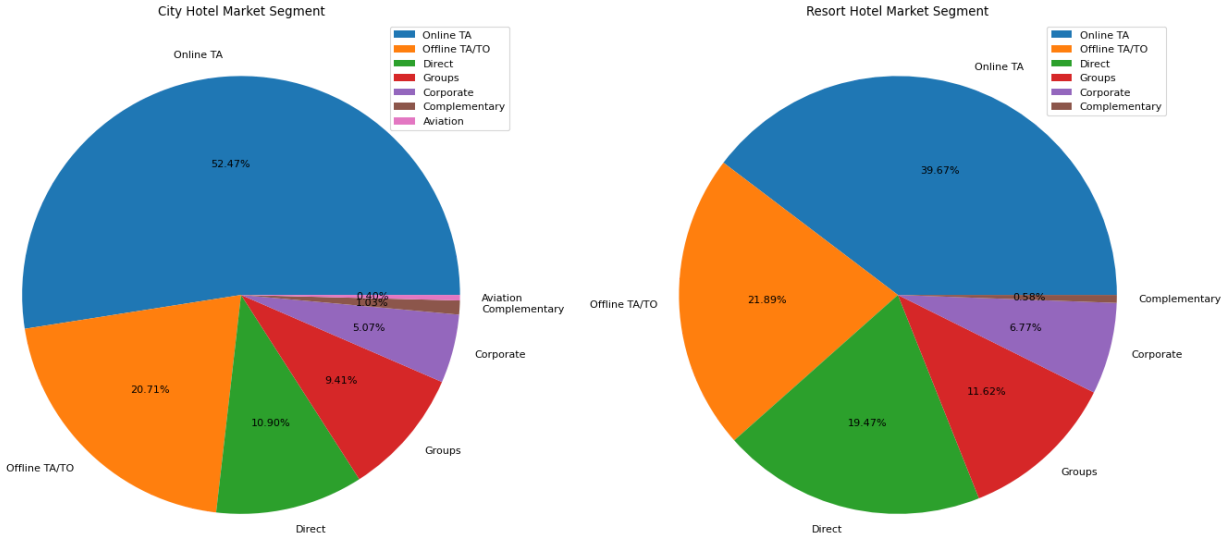


The main customer type of the hotel is transient travelers, accounting for about 70%.

1. Hotel booking method

In [13]:

```
#create pie charts to visualize the distribution of market segments for both City Hotel
city_segment=city_hotel.market_segment.value_counts()
resort_segment=resort_hotel.market_segment.value_counts()
plt.figure(figsize=(21,12),dpi=80)
plt.subplot(1,2,1)
plt.pie(city_segment,labels=city_segment.index,autopct='%.2f%%')
plt.legend()
plt.title('City Hotel Market Segment')
plt.subplot(1,2,2)
plt.pie(resort_segment,labels=resort_segment.index,autopct='%.2f%%')
plt.title('Resort Hotel Market Segment')
plt.legend()
plt.show()
```

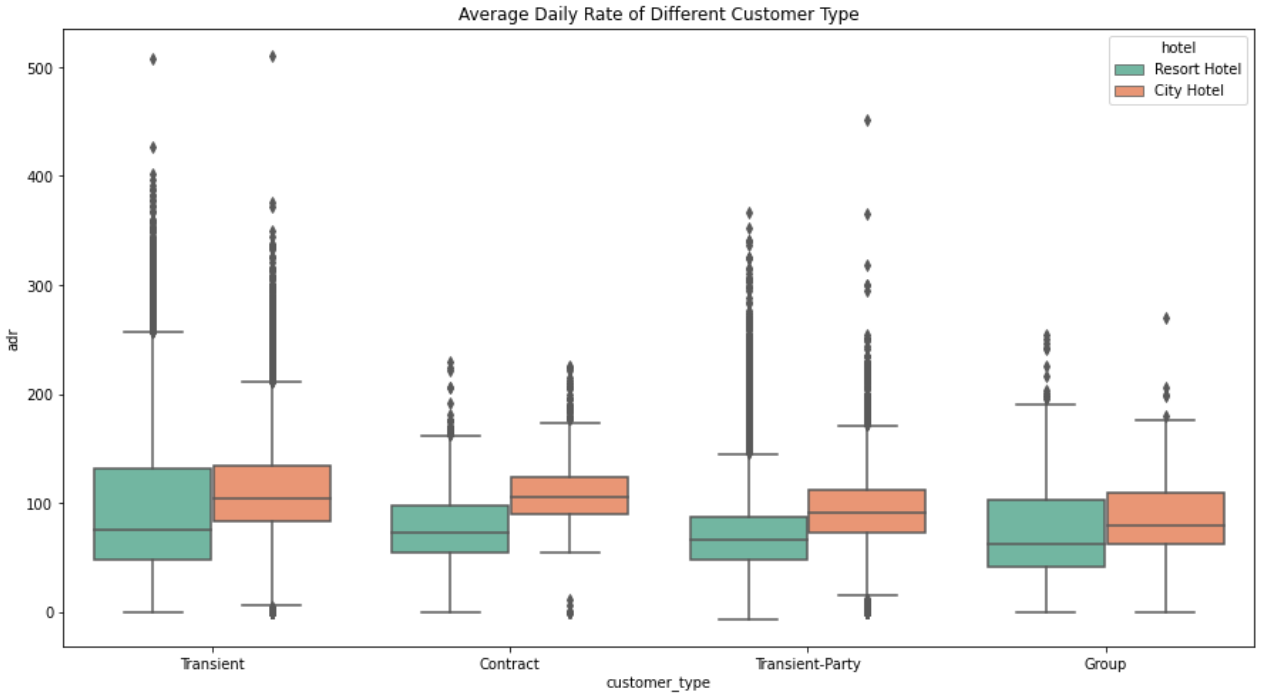


The customers of the two hotels mainly come from online travel agencies, which account for even more than 50% of the City Hotel; offline travel agencies come next, accounting for about 20%.

1. Average daily expenses of various types of passengers

In [14]:
```
#visualize the distribution of Average Daily Rate (adr) across different customer types
plt.figure(figsize=(15,8))
sns.boxplot(x='customer_type'
            ,y='adr'
            ,hue='hotel'
            ,data=df[df.is_canceled==0]
            ,palette=sns.color_palette('Set2',2)
           )
plt.title('Average Daily Rate of Different Customer Type')
```

Out[14]:
```
Text(0.5, 1.0, 'Average Daily Rate of Different Customer Type')
```

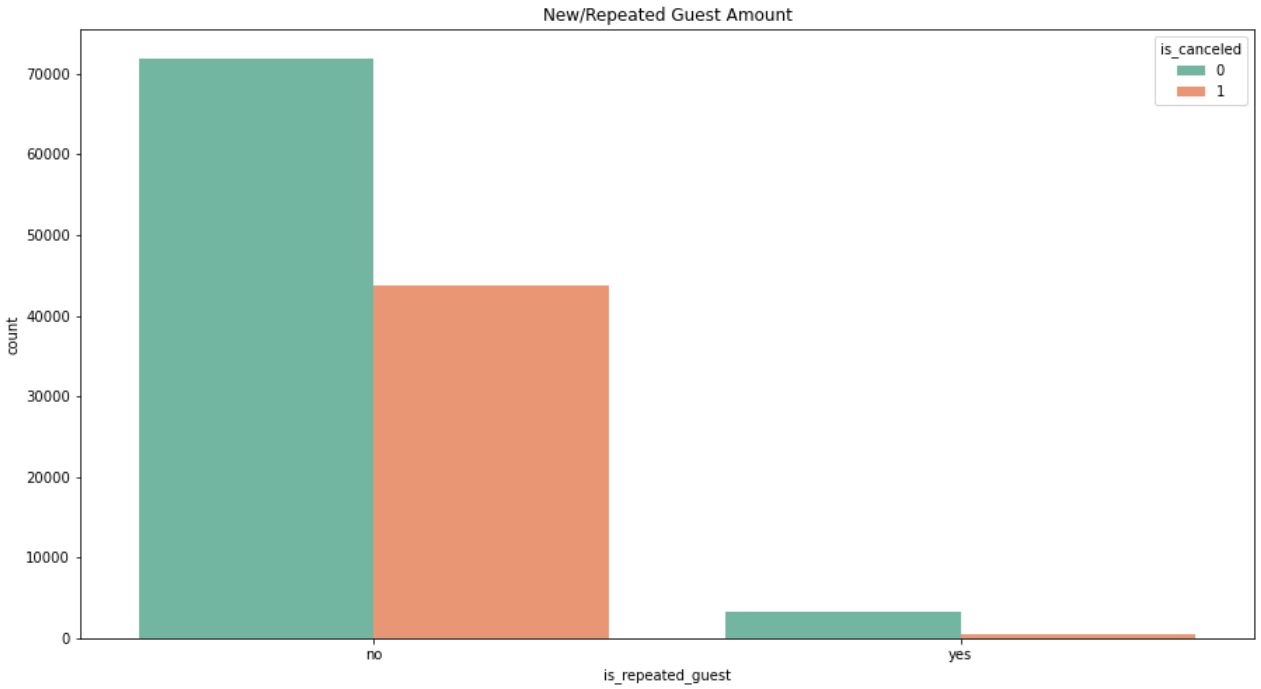


The average daily expenditure of all types of customers of City Hotel is higher than that of Resort Hotel; among the four types of customers, the consumption of individual travelers (Transient) is the highest and that of group travelers (Group) is the lowest.

7.Number of new and old customers and cancellation rate

In [15]:
```
# visualize the count of bookings, categorized by whether the guest is a repeated guest
plt.figure(figsize=(15,8))
sns.countplot(x='is_repeated_guest'
              ,data=df
              ,hue='is_canceled'
              ,palette=sns.color_palette('Set2',2)
             )
plt.title('New/Repeated Guest Amount')
plt.xticks(range(2),['no','yes'])
```

Out[15]:
```
([<matplotlib.axis.XTick at 0x1a663577a60>,
  <matplotlib.axis.XTick at 0x1a663577a30>],
 [Text(0, 0, 'no'), Text(1, 0, 'yes')])
```



In [16]:
```
#calculate and printing the cancellation rates for new and repeated guests
guest_cancel=(df.loc[df['is_canceled']==1]['is_repeated_guest'].value_counts()/df['is_r
guest_cancel.index=['New Guest', 'Repeated Guest']
print('Cancellation rate for new and old customers'.center(15),guest_cancel,sep='\n')
```

```
Cancellation rate for new and old customers
New Guest         0.377851
Repeated Guest    0.144882
Name: is_repeated_guest, dtype: float64
```

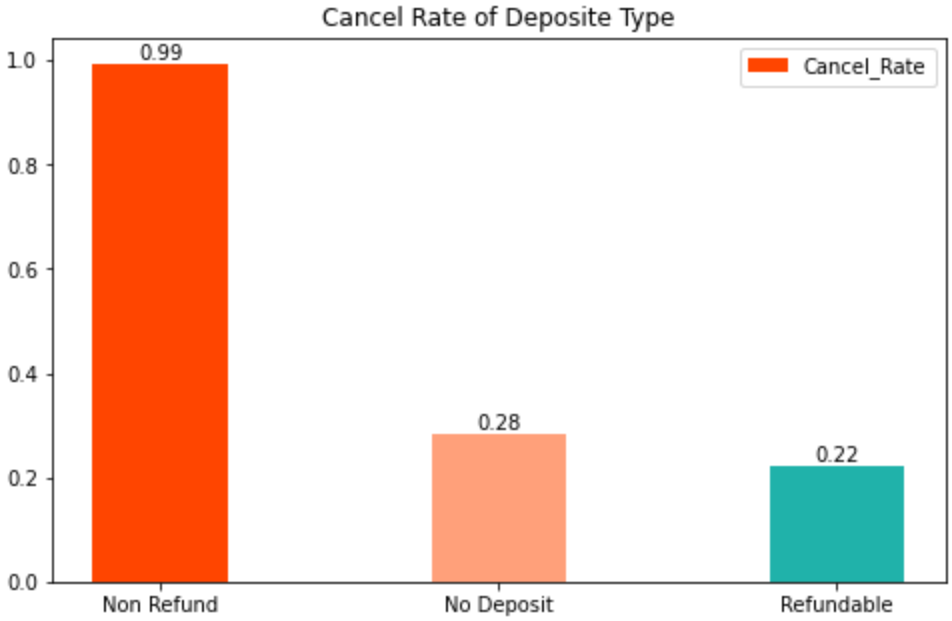
The cancellation rate for regular customers was 14.4%, while the cancellation rate for new customers reached 37.8%, which was 24 percentage points higher than that for regular customers.

1. Deposit method and reservation cancellation rate

In [17]:
```
print('Three deposit methods for booking quantity'.center(15),df['deposit_type'].value_
```

```
Three deposit methods for booking quantity
No Deposit    104641
Non Refund     14587
Refundable       162
Name: deposit_type, dtype: int64
```

In [18]:
```
#calculate the cancellation rates based on the 'deposit_type', and visualizing these ra
deposit_cancel=(df.loc[df['is_canceled']==1]['deposit_type'].value_counts()/df['deposit
plt.figure(figsize=(8,5))
x=range(len(deposit_cancel.index))
y=deposit_cancel.values
plt.bar(x,y,label='Cancel_Rate',color=['orangered','lightsalmon','lightseagreen'],width
plt.xticks(x,deposit_cancel.index)
plt.legend()
plt.title('Cancel Rate of Deposite Type')
for x,y in zip(x,y):
    plt.text(x,y,'%.2f' % y,ha = 'center',va = 'bottom')
```

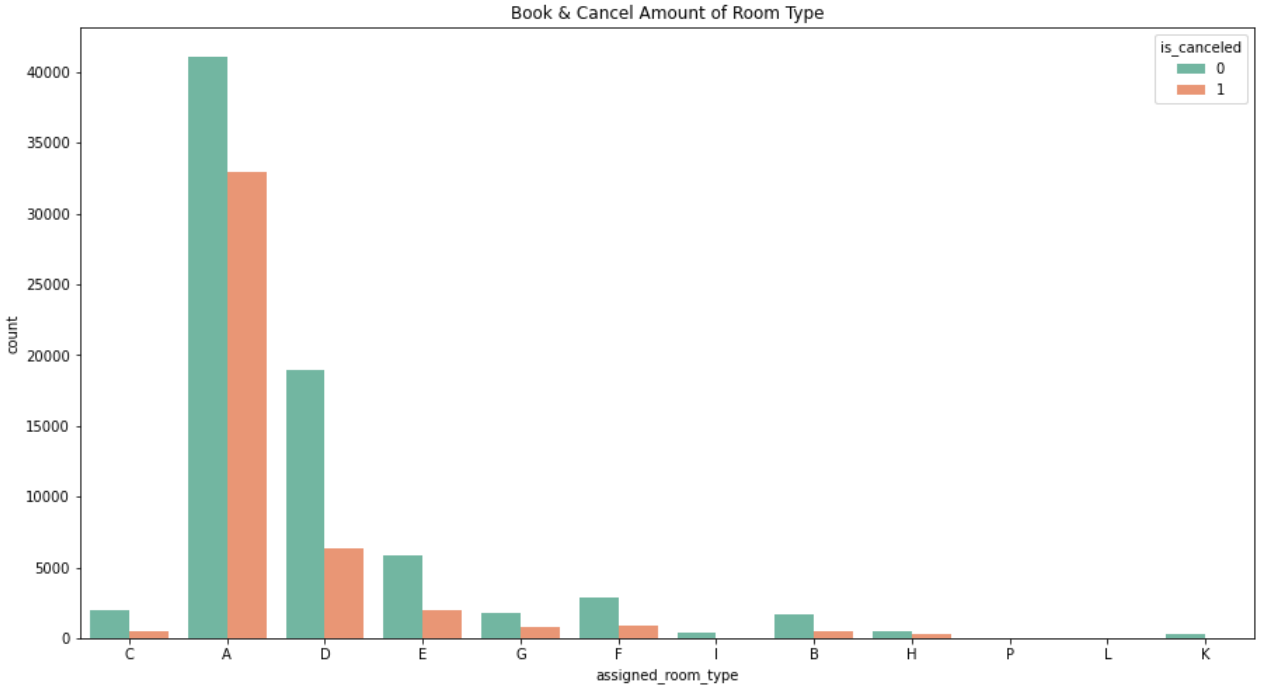


'No Deposit' is the method with the highest number of bookings and has a low cancellation rate, while the cancellation rate of non-refundable type is as high as 99%. This type of deposit method can be reduced to reduce Customer cancellation rate.

1. Room type and cancellation volume

In [19]:
```
#visualize the counts of bookings and cancellations across different assigned room type
plt.figure(figsize=(15,8))
sns.countplot(x='assigned_room_type'
              ,data=df
              ,hue='is_canceled'
              ,palette=sns.color_palette('Set2',2)
             )
plt.title('Book & Cancel Amount of Room Type')
```

Out[19]: Text(0.5, 1.0, 'Book & Cancel Amount of Room Type')



In [20]:
```
#calculate cancellation rates for the top 7 assigned room types and printing them in de
room_cancel=df.loc[df['is_canceled']==1]['assigned_room_type'].value_counts()[:7]/df['a
print('Cancellation rates for different room types'.center(5),room_cancel.sort_values(a
```

```
Cancellation rates for different room types
A    0.444925
G    0.305523
E    0.252114
D    0.251244
F    0.247134
B    0.236708
C    0.187789
Name: assigned_room_type, dtype: float64
```

Among the top seven room types with the most bookings, the cancellation rates of room types A and G are higher than other room types, and the cancellation rate of room type A is as high as 44.5%.

Conclusion

1. The booking volume and cancellation rate of City Hotel are much higher than that of Resort Hotel. The hotel should conduct customer surveys to gain an in-depth understanding of the factors that cause customers to give up on bookings in order to reduce customer cancellation rates.

2. Hotels should make good use of the peak tourist season of July and August every year. They can increase prices appropriately while ensuring service quality to obtain more profits, and conduct preferential activities during the off-season (winter), such as Christmas sales and New Year activities, to reduce Hotel vacancy rate.

3. Hotels need to analyze customer profiles from major source countries such as Portugal and the United Kingdom, understand the attribute tags, preferences and consumption characteristics of these customers, and launch exclusive services to reduce customer cancellation rates.

4. Since individual travelers are the main customer group of hotels and have high consumption levels, hotels can increase the promotion and marketing of independent travelers through online and offline travel agencies, thereby attracting more tourists of this type.

5. The cancellation rate of new customers is 24% higher than that of old customers. Therefore, hotels should focus on the booking and check-in experience of new customers, and provide more guidance and benefits to new customers, such as providing discounts to first-time customers and conducting research on new customers. Provide feedback on satisfaction and dissatisfaction with your stay to improve future services and maintain good old customers.

6. The cancellation rate of non-refundable deposits is as high as 99%. Hotels should optimize this method, such as returning 50% of the deposit, or cancel this method directly to increase the occupancy rate.

7. The cancellation rate of room types A and G is much higher than that of other room types. The hotel should carefully confirm the room information with the customer when making a reservation, so that the customer can fully understand the room situation, avoid cognitive errors, and at the same time be able to understand the room facilities. Optimize and improve service levels.

Data Processing (Week 4)

In [21]:
```
#create a new DataFrame 'df1' from 'df'
df1=df.drop(labels=['reservation_status_date'],axis=1)
```

Handling Categorical Variables

In [22]:
```
cate=df1.columns[df1.dtypes == "object"].tolist() #getting the names of all columns in
#categorical variables expressed as numbers
num_cate=['agent','company','is_repeated_guest']
cate=cate+num_cate
```

In [23]:
```
import numpy as np #linear algebra
#creating a dictionary
results={}
for i in ['agent','company']:
    result=np.sort(df1[i].unique())
    results[i]=result
results
```

Out[23]:
```
{'agent': array([  1.,   2.,   3.,   4.,   5.,   6.,   7.,   8.,   9.,  10.,  11.,
         12.,  13.,  14.,  15.,  16.,  17.,  19.,  20.,  21.,  22.,  23.,
         24.,  25.,  26.,  27.,  28.,  29.,  30.,  31.,  32.,  33.,  34.,
         35.,  36.,  37.,  38.,  39.,  40.,  41.,  42.,  44.,  45.,  47.,
```

```
        50.,  52.,  53.,  54.,  55.,  56.,  57.,  58.,  59.,  60.,  61.,
        63.,  64.,  66.,  67.,  68.,  69.,  70.,  71.,  72.,  73.,  74.,
        75.,  77.,  78.,  79.,  81.,  82.,  83.,  85.,  86.,  87.,  88.,
        89.,  90.,  91.,  92.,  93.,  94.,  95.,  96.,  98.,  99., 103.,
       104., 105., 106., 107., 110., 111., 112., 114., 115., 117., 118.,
       119., 121., 122., 126., 127., 128., 129., 132., 133., 134., 135.,
       138., 139., 141., 142., 143., 144., 146., 147., 148., 149., 150.,
       151., 152., 153., 154., 155., 156., 157., 158., 159., 162., 163.,
       165., 167., 168., 170., 171., 173., 174., 175., 177., 179., 180.,
       181., 182., 183., 184., 185., 187., 191., 192., 193., 195., 196.,
       197., 201., 205., 208., 210., 211., 213., 214., 215., 216., 219.,
       220., 223., 227., 229., 232., 234., 235., 236., 240., 241., 242.,
       243., 244., 245., 247., 248., 249., 250., 251., 252., 253., 254.,
       256., 257., 258., 261., 262., 265., 267., 269., 270., 273., 275.,
       276., 278., 280., 281., 282., 283., 285., 286., 287., 288., 289.,
       290., 291., 294., 295., 296., 298., 299., 300., 301., 302., 303.,
       304., 305., 306., 307., 308., 310., 313., 314., 315., 321., 323.,
       324., 325., 326., 327., 328., 330., 331., 332., 333., 334., 335.,
       336., 337., 339., 341., 344., 346., 348., 350., 352., 354., 355.,
       358., 359., 360., 363., 364., 367., 368., 370., 371., 375., 378.,
       384., 385., 387., 388., 390., 391., 393., 394., 397., 403., 404.,
       405., 406., 408., 410., 411., 414., 416., 418., 420., 423., 425.,
       426., 427., 429., 430., 431., 432., 433., 434., 436., 438., 440.,
       441., 444., 446., 449., 450., 451., 453., 454., 455., 459., 461.,
       464., 467., 468., 469., 472., 474., 475., 476., 479., 480., 481.,
       483., 484., 492., 493., 495., 497., 502., 508., 509., 510., 526.,
       527., 531., 535.,  nan]),
'company': array([  6.,   8.,   9.,  10.,  11.,  12.,  14.,  16.,  18.,  20.,  22.,
        28.,  29.,  31.,  32.,  34.,  35.,  37.,  38.,  39.,  40.,  42.,
        43.,  45.,  46.,  47.,  48.,  49.,  51.,  52.,  53.,  54.,  59.,
        61.,  62.,  64.,  65.,  67.,  68.,  71.,  72.,  73.,  76.,  77.,
        78.,  80.,  81.,  82.,  83.,  84.,  85.,  86.,  88.,  91.,  92.,
        93.,  94.,  96.,  99., 100., 101., 102., 103., 104., 105., 106.,
       107., 108., 109., 110., 112., 113., 115., 116., 118., 120., 122.,
       126., 127., 130., 132., 135., 137., 139., 140., 142., 143., 144.,
       146., 148., 149., 150., 153., 154., 158., 159., 160., 163., 165.,
       167., 168., 169., 174., 178., 179., 180., 183., 184., 185., 186.,
       192., 193., 195., 197., 200., 202., 203., 204., 207., 209., 210.,
       212., 213., 215., 216., 217., 218., 219., 220., 221., 222., 223.,
       224., 225., 227., 229., 230., 232., 233., 234., 237., 238., 240.,
       242., 243., 245., 246., 250., 251., 253., 254., 255., 257., 258.,
       259., 260., 263., 264., 268., 269., 270., 271., 272., 273., 274.,
       275., 277., 278., 279., 280., 281., 282., 284., 286., 287., 288.,
       289., 290., 291., 292., 293., 297., 301., 302., 304., 305., 307.,
       308., 309., 311., 312., 313., 314., 316., 317., 318., 319., 320.,
       321., 323., 324., 325., 329., 330., 331., 332., 333., 334., 337.,
       338., 341., 342., 343., 346., 347., 348., 349., 350., 351., 352.,
       353., 355., 356., 357., 358., 360., 361., 362., 364., 365., 366.,
       367., 368., 369., 370., 371., 372., 373., 376., 377., 378., 379.,
       380., 382., 383., 384., 385., 386., 388., 390., 391., 392., 393.,
       394., 395., 396., 397., 398., 399., 400., 401., 402., 403., 405.,
       407., 408., 409., 410., 411., 412., 413., 415., 416., 417., 418.,
       419., 420., 421., 422., 423., 424., 425., 426., 428., 429., 433.,
       435., 436., 437., 439., 442., 443., 444., 445., 446., 447., 448.,
       450., 451., 452., 454., 455., 456., 457., 458., 459., 460., 461.,
       465., 466., 470., 477., 478., 479., 481., 482., 483., 484., 485.,
       486., 487., 489., 490., 491., 492., 494., 496., 497., 498., 499.,
       501., 504., 506., 507., 511., 512., 513., 514., 515., 516., 518.,
       520., 521., 523., 525., 528., 530., 531., 534., 539., 541., 543.,
        nan])}
```

In [24]:
```
# the agent and company columns have a large number of empty values and no 0 values, so
df1[['agent','company']]=df1[['agent','company']].fillna(0,axis=0)
```

In [25]:
```
df1.loc[:,cate].isnull().mean()
```

Out[25]:
```
hotel                  0.000000
arrival_date_month     0.000000
meal                   0.000000
country                0.004087
market_segment         0.000000
distribution_channel   0.000000
reserved_room_type     0.000000
assigned_room_type     0.000000
deposit_type           0.000000
customer_type          0.000000
reservation_status     0.000000
name                   0.000000
email                  0.000000
phone-number           0.000000
credit_card            0.000000
agent                  0.000000
company                0.000000
is_repeated_guest      0.000000
dtype: float64
```

In [26]:
```
#create new variables in_company and in_agent to classify passengers. If company and ag
df1.loc[df1['company'] == 0,'in_company']='NO'
df1.loc[df1['company'] != 0,'in_company']='YES'
df1.loc[df1['agent'] == 0,'in_agent']='NO'
df1.loc[df1['agent'] != 0,'in_agent']='YES'
```

In [27]:
```
#create a new feature same_assignment. If the booked room type is consistent with the a
df1.loc[df1['reserved_room_type'] == df1['assigned_room_type'],'same_assignment']='Yes'
df1.loc[df1['reserved_room_type'] != df1['assigned_room_type'],'same_assignment']='No'
```

In [28]:
```
#delete four features except 'reserved_room_type', 'assigned_room_type', 'agent', 'comp
df1=df1.drop(labels=['reserved_room_type','assigned_room_type','agent','company'],axis=
```

In [29]:
```
#reset 'is_repeated_guest', frequent guests are marked as YES, non-repeated guests are
df1['is_repeated_guest'][df1['is_repeated_guest']==0]='NO'
df1['is_repeated_guest'][df1['is_repeated_guest']==1]='YES'
```

In [30]:
```
#filling the missing values in the 'country' column of the DataFrame 'df1' with the mod
df1['country']=df1['country'].fillna(df1['country'].mode()[0])
```

In [31]:
```
for i in ['in_company','in_agent','same_assignment']:
    cate.append(i)

for i in ['reserved_room_type','assigned_room_type','agent','company']:
```

```
    cate.remove(i)
cate
```

Out[31]:
```
['hotel',
 'arrival_date_month',
 'meal',
 'country',
 'market_segment',
 'distribution_channel',
 'deposit_type',
 'customer_type',
 'reservation_status',
 'name',
 'email',
 'phone-number',
 'credit_card',
 'is_repeated_guest',
 'in_company',
 'in_agent',
 'same_assignment']
```

In [32]:
```
#encoding categorical features
from sklearn.preprocessing import OrdinalEncoder
oe = OrdinalEncoder()
oe = oe.fit(df1.loc[:,cate])
df1.loc[:,cate] = oe.transform(df1.loc[:,cate])
```

Working With Continuous Variables

In [33]:
```
#to filter out continuous variables, you need to delete the label 'is_canceled' first.
col=df1.columns.tolist()
col.remove('is_canceled')
for i in cate:
    col.remove(i)
col
```

Out[33]:
```
['lead_time',
 'arrival_date_year',
 'arrival_date_week_number',
 'arrival_date_day_of_month',
 'stays_in_weekend_nights',
 'stays_in_week_nights',
 'adults',
 'children',
 'babies',
 'previous_cancellations',
 'previous_bookings_not_canceled',
 'booking_changes',
 'days_in_waiting_list',
 'adr',
 'required_car_parking_spaces',
 'total_of_special_requests']
```

In [34]:
```
df1[col].isnull().sum()
```

Out[34]:
```
lead_time                       0
arrival_date_year               0
```

```
arrival_date_week_number          0
arrival_date_day_of_month         0
stays_in_weekend_nights           0
stays_in_week_nights              0
adults                            0
children                          4
babies                            0
previous_cancellations            0
previous_bookings_not_canceled    0
booking_changes                   0
days_in_waiting_list              0
adr                               0
required_car_parking_spaces       0
total_of_special_requests         0
dtype: int64
```

In [35]:
```
#use mode to fill null values in xtrain children column
df1['children']=df1['children'].fillna(df1['children'].mode()[0])
```

In [36]:
```
#continuous variables are dimensionless
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
ss = ss.fit(df1.loc[:,col])
df1.loc[:,col] = ss.transform(df1.loc[:,col])
```

Correlation Coefficient of Each Variable

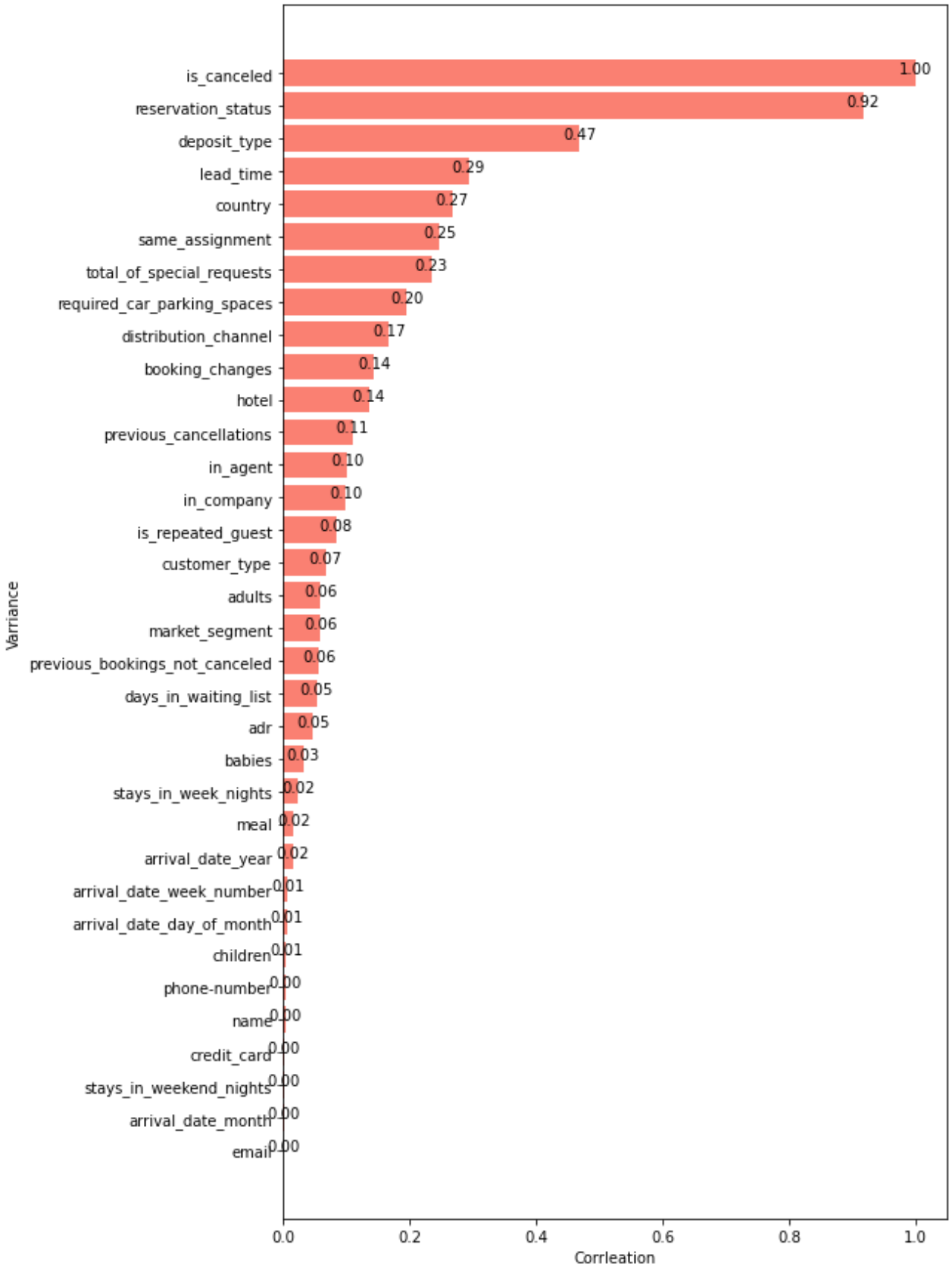
In [37]:
```
#calculating the correlation of all numerical columns with the 'is_canceled column' in
cor=df1.corr()
cor=abs(cor['is_canceled']).sort_values()
cor
```

Out[37]:
```
email                             0.000723
arrival_date_month                0.001491
stays_in_weekend_nights           0.001791
credit_card                       0.002515
name                              0.004253
phone-number                      0.004342
children                          0.005036
arrival_date_day_of_month         0.006130
arrival_date_week_number          0.008148
arrival_date_year                 0.016660
meal                              0.017678
stays_in_week_nights              0.024765
babies                            0.032491
adr                               0.047557
days_in_waiting_list              0.054186
previous_bookings_not_canceled    0.057358
market_segment                    0.059338
adults                            0.060017
customer_type                     0.068140
is_repeated_guest                 0.084793
in_company                        0.099310
in_agent                          0.102068
previous_cancellations            0.110133
hotel                             0.136531
```

```
booking_changes                 0.144381
distribution_channel            0.167600
required_car_parking_spaces     0.195498
total_of_special_requests       0.234658
same_assignment                 0.247770
country                         0.267502
lead_time                       0.293123
deposit_type                    0.468634
reservation_status              0.917196
is_canceled                     1.000000
Name: is_canceled, dtype: float64
```

In [38]:

```
#create a horizontal bar plot using Matplotlib to visualize the absolute correlation va
plt.figure(figsize=(8,15))
x=range(len(cor.index))
name=cor.index
y=abs(cor.values)
plt.barh(x,y,color='salmon')
plt.yticks(x,name)
for x,y in zip(x,y):
    plt.text(y,x-0.1,'%.2f' % y,ha = 'center',va = 'bottom')
plt.xlabel('Corrleation')
plt.ylabel('Varriance')
plt.show()
```



The reservation status ('reservation_status') has the highest correlation with whether to cancel the reservation, reaching 0.92, but considering that it may cause the model to overfit in the future, it is deleted; the deposit type ('deposit_type') reaches 0.47, creating a characteristic Whether the reservation and assigned room type are consistent ('same_assignment') also has a correlation of 0.25.

In [39]:

```
#copy 'df1' with the column labeled 'reservation_status' dropped.
df2=df1.drop('reservation_status',axis=1)
```